College of Information Science and Technology

Published in: Proceedings of BCS Specialist Group on IS Methodologies, 3rd Annual Conference, *The Application of Methodologies in Industrial and Business Change*, North East Wales Institute, Wrexham, UK - September 1995

THE ROLE OF METHODOLOGIES IN IT-RELATED ORGANISATIONAL CHANGE

Susan Gasson
Information Systems Research Unit, Warwick Business School
University of Warwick, Coventry, West Midlands CV4 7AL
Tel: **01203-524582,** ***E-mail:*** **orssg@razor.warwick.ac.uk**

Key Words: Methodology, Information Systems Development

Abstract

Despite much academic interest, little is really known about information systems development and the use of systems development methodologies. There has been little evaluation of methodologies in use and only limited research into their selection, adaptation and use, in practice. This paper relates theoretical work to empirical studies on these issues and discusses the implications for research and practice.

Introduction

The research area of information systems development methodologies could almost be classified as a Site of Special Scientific Interest under UK conservation laws. Not only does this area include many endangered species, but productivity sightings are often made and seldom proven. This paper attempts an overview of what is known about methodologies and their role in IT-related change in organisations.

Academic interest in the area of systems development methodologies appears to have peaked in the early 1980s. IFIP WG8.1 initiated a series of conferences on information systems design methodologies (Olle et al., 1982, 1983, 1986). However, the concept of "methodology" used for these conferences was limited to the design stage of the system development life-cycle and many of the methodologies analysed were derived from academic research rather than practice. Lyytinen (1987) criticises the methodologies discussed in Olle et al. (1982), arguing that few of them provide "a penetrating definition of the IS, its parts and purpose". Lyytinen further argues that development methodologies have an inadequate conception of the phenomena IS developers confront in practice and that their assumptions about the nature of systems development conflict with empirical findings.

In a narrow sense, information systems development may be seen as technical change. A single problem is seen as well-defined, a technical solution is proposed, evaluated and implemented. However, Klein and Hirschheim (1987) suggest that a societal change is taking place, where IS development is seen as involving much broader social and organisational change. Information systems development is seen as the derivation of a new social system, *supported by* computer-based technology (Land & Hirschheim, 1983). Alternative development approaches, such as Soft Systems Methodology (Checkland, 1981), Evolutionary Development (Eason, 1982), the ETHICS approach (Mumford, 1983), Multiview (Avison & Wood-Harper, 1990) and Change Analysis (Goldkuhl & Rostlinger, 1993) have been developed to support the wider scope required by such a definition of IS development. These approaches concentrate upon development methodologies which are largely contingent upon organisational context and upon reflective (usually academic) practitioners; they do not provide alternative models for the processes engaged in by IS professionals with limited time and resources.

To understand the options open to us, when proposing changes in practice, we need to understand what that practice *is*: to investigate the role played by methodologies in IT-related change. This paper presents an investigation into that role by examining recent empirical evidence to answer three questions:

1. What methodologies are in use?
2. How are systems development methodologies selected?
3. To what extent do methodologies support systems development?

Before addressing these questions, it is useful to examine what is meant by the term 'information systems development methodology'.

What Is An Information Systems Development Methodology?

There has been some debate about whether the term 'methodology', which literally means "the study of methods" can be used to refer to a particular methodological approach to information systems development. Jayaratna (1994) emphasises that a methodology provides an "explicit way of structuring" systems development:

> "Methodologies contain models and reflect particular perspectives of 'reality' based on a set of philosophical paradigms. A methodology should tell you 'what' steps to take and 'how' to perform those steps but most importantly the reasons 'why' those steps should be taken, in that particular order." (*Jayaratna, 1994, pg. 37).*

Maddison et al. (1984) define a methodology as "a recommended collection of philosophies, phases, procedures, rules, techniques, tools, documentation, management and training for developers of information systems", while Avison & Fitzgerald (1988) state that:

> "A methodology is a collection of procedures, techniques, tools and documentation aids which will help the systems developers in their efforts to implement a new information system. A methodology will consist of phases, themselves consisting of sub-phases, which will guide the system developers in their choice of the techniques that might be appropriate at each stage of the project and also help them plan, manage, control and evaluate information systems projects". (*Avison & Fitzgerald, 1988, pg. 4).*

In the sense given in each of these definitions, a methodology is more than just a method (the 'how' of information systems development), or a process-model. A methodology is an holistic *approach*: it embodies an analytical framework which is conveyed through intersubjective representational practices and operationalised through a 'toolbox' of analytical methods, tools and techniques. Underlying the analytical framework is a process-model which indicates the sequence and relative duration of development activities. Standardised organisational procedures, such as user-participation mechanisms or formal review meetings, are critical to an holistic definition of a methodology, so that development outputs may conform to pre-defined standards, may be governed by pre-defined expectations and so that there may be a unified system of control over all IS development projects which use the methodology. Underlying all of these elements is a philosophical basis, which justifies the need for each element of the methodology and ensures intersubjectivity and commitment between development team members.

The elements of a methodology are illustrated in Figure 1. These elements permit an individual to structure their understanding of appropriate solutions for a problem situation, according to their perspective and their previous experience of both the problem context and the methodology. A methodology affects the way in which individuals' will perceive the context and tasks of development, with each component layer of the methodology acting as a filter to the next layer. Ultimately, the problem situation is perceived through the filters provided by successive elements of the methodology; these elements in turn are filtered through stakeholders' perceptions of their utility and application. Thus stakeholders' perceptions of the

representational techniques which they use will be coloured by the organisational procedures and standards which provide the context for their use. For example, a client may perceive a problem-situation as very complex when viewed through the filters of a "hard" systems philosophy, rigid waterfall process-model, data-analysis, entity-relationship diagrams and formal user-validation meetings, but might view the same problem-situation as much less complex when viewed through the filters of a user-centred philosophy, evolutionary development, activity-modelling, SSM conceptual models and full user-participation in development.

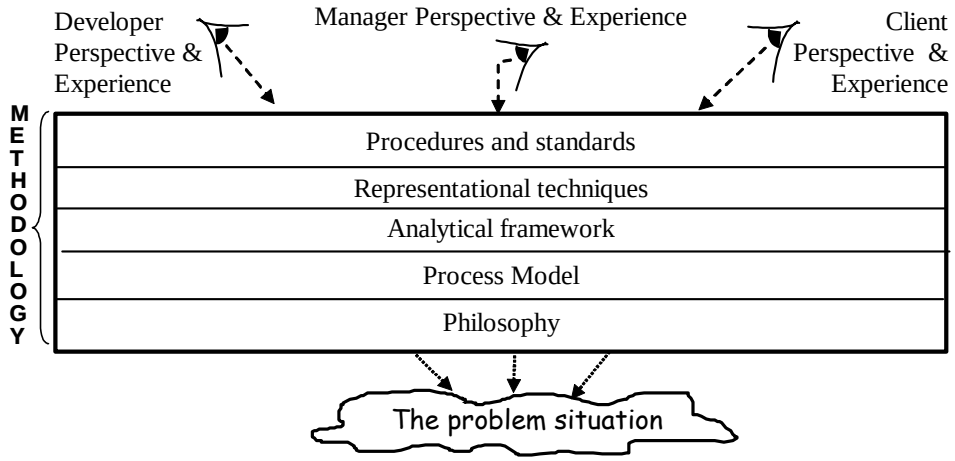

Figure 1: The Elements of a Methodology

Different people may have very different understandings of a methodology, depending upon their background and their experience of this type of methodology. In the context of this scope of meaning, the use of the term methodology in the wide sense used by development practitioners does not seem inappropriate, especially in the context of a body of academic research which has as its objective the improvement of information systems development practice; to use the term "method" does.

Methodologies may differ widely in terms of their philosophy, objectives and system modelling approaches. Avison (1990) classifies methodologies as pertaining to seven different approaches: systems approaches, planning approaches, participative approaches, prototyping approaches, automated approaches, structured approaches and data approaches. Avison's classification and descriptions have been used to generate Table 1, which illustrates differences between various types of methodologies. The classification cannot be said to be exhaustive. Missing, for example, is the traditional approach, classified by Boehm et al. (1984) as the "specifying" approach: as requirements are specified (it is assumed, completely) before the design phase begins. The traditional approach uses the waterfall model to manage the process but its support of design processes is unstructured and does not use the decompositional tools and techniques of structured methodologies.

There is confusion in many studies about the definition of various methodologies: while Avison (1990) sees structured approaches as distinct from the data approach (centred upon data structures and the use of 'data dictionaries'), Sumner & Sitek (1986) note that structured systems analysis combines process design and data design, using such tools as a data dictionary. Traditional methodologies are often confused and/or combined with structured methodologies, because they use the same process-model. In the context of this paper, the term 'structured methodology' is used to mean any methodological approach which is based upon structured decomposition of system requirements and managed using the waterfall model of the system

development life-cycle. The traditional approach is not seen as using a development methodology in the wide sense discussed above, but as an approach to project management and control.

Table 1: Classification of IS Development Methodologies By Development Process
(developed from Avison, 1990)

Approach	**Perspective/Philosophy**	**The approach addresses:**	**The approach neglects:**
Soft systems approach	Organisational problems are ill-defined. Philosophy is to understand the organisation holistically, analysing the structure of organisations from many perspectives.	Organisational 'problem' definition. Information system objectives and requirements analysis.	Later stages of the systems development life-cycle (SDLC).
Planning approaches	IS development should be aligned with strategic business needs.	Pre-planning involved in developing information systems.	Actual operational needs of the development project.
Participative approaches	Effective IS development is based on emancipatory organisational change.	User appreciation and determination of system operation and limitations.	Wider organisational implications, such as structural changes.
Prototyping approaches	*Varies*: prototyping may be used for emancipatory or experimental, technical purposes. May be one-off or evolutionary.	User and/or developer appreciation of technical system operation and limitations.	Concentrates on local system, neglecting wider organisational implications, such as structural changes.
Automated approaches	Problems in IS development lie in imprecise definition of technical system and lack of accurate documentation.	Early definition of system requirements. Rapid production of system design & implementation from specified requirements.	Human, social and organisational impacts of system.
Structured approaches	A well-defined organisational 'problem' forms the basis for the new system. There is 'one best way' of approaching development	Early definition of system requirements. Standardisation & co-ordination of development tasks.	Human, social and organisational impacts of system.
Data approaches	The organisation can be understood through an analysis of its use of information.	Use of technology to support existing or required data-flows and transactions.	Human, social and organisational impacts of system.

Saarinen (1990) classifies methodologies by four variables:

(i) the system development strategy: linear (waterfall model), mixed methodology, or prototyping (by which he means evolutionary development)
(ii) software development tools: software package, third-generation programming language or application generator (4GL)
(iii) the formality of the system development method
(iv) the level of planning and management control in the project.

Saarinen's (1990) classification is useful for detailed analysis of development approaches, but concentrates as much on the management of the methodology as on the type of methodology. Whilst this is essential for an understanding of *how* methodologies are used, a more useful perspective for *what* approaches are in use might be to classify methodologies by their main objective. This has been attempted in Table 2, which concentrates upon development strategies, rather than the tools and methods which are used to support individual tasks.

Table 2: Classification of Methodologies By Development Strategy

Approach	Perspective/Philosophy	The approach addresses:	The approach neglects:
Project management	The main problem of IS development is lack of control over the process.	Standardisation & co-ordination of development tasks.	Human, social and organisational impacts of system.
Risk Management	Understanding of system requirements at start of project is imperfect and needs to be redefined as objectives become apparent.	Evaluation of fit between current task outcomes and latest understanding of system requirements.	Financial/resource focus - neglects human, social and organisational impacts of system.
Business process re-design	Existing organisational structures impede business effectiveness. "Throw it all away & start again".	The design of all levels of business activities and process, from strategic to operational.	The detailed development of physical systems.
Software re-use	Efficiency is best maintained through not reinventing the wheel each time a new system is developed.	Production of software libraries, from which system components selected	Human, social and organisational impacts of system.
Software Engineering	A technical solution can be optimised by use of formal notations to precisely define system operations.	Systems where the problem is well-defined and a technical solution is appropriate.	Human, social and organisational impacts of system.
Socio-technical design	Emancipatory: social, task and organisational aspects of the information system need to be optimised jointly with technical aspects.	Design of work and social system and specification of technical support system requirements.	*Appropriate* design and evaluation of technical system. Relies on practitioner expertise as much as methodology.
Evolutionary development	Rapid-cycle iteration in development can effectively incorporate feedback from users on system impact on organisation and work tasks.	Organisational fit of system and adaptation to change. Relies on the use of RAD, 4GL, or prototyping tools.	Financial and management control of processes. May be used experimentally or for emancipation.
Computer-Aided Software Engineering	An information system may be defined through its data-structures. Given suitable tools, production of technical systems can be automated.	Precise analysis and design of data models to support organisation information flows.	Human, social and organisational impacts of system.

A classification of methodologies does not tell us much about organisational practice: only about what academics and consultants, the originators of most methodologies, think is desirable for organisational practice. Jeffries et al. (1981) argue that the professional literature in the field of software design (by which they mean methodology guidelines) ought to relate to accepted standards of good practice as the formalised methodologies were written by experts trying to convey to others the procedures they used to perform the task. However, empirical research in organisational contexts (c.f. Jenkins et al., 1984; Curtis et al., 1988; Hornby et al., 1992; Davidson, 1993) shows that methodologies do not represent a 'theory-in-use', but a 'theory-of-action' (Argyris and Schon, 1978): they represent a rule-based interpretation of *what should be done*, rather than what people actually *do*.

What Methodologies Are In Use?

Empirical studies of methodology diffusion are rare - the practical problems of studying the use of methodologies in organisations mean that few researchers undertake this type of study. In the

US, Wynekoop & Russo (1993) report a survey which indicates that structured methodologies (based upon the detailed specification and structured decomposition of documented system requirements) were in use in about two-thirds of their sample organisations. However, Jenkins et al. (1984) indicate that 43% of users of a commercial systems development methodology and 50% of users of methodologies developed in-house felt that their methodology was primarily a project management aid - controlling the process through staged division of tasks - rather than an aid to system development. This emphasis reflects a traditional, rather than a structured approach. This finding may have occurred because the study was conducted at a time when the "software crisis" was perceived to be mainly about managing the development process (Friedman and Cornford, 1989). Once process management was perceived to be under control, the crisis moved on, to become a concern with user-relations, which may explain the preoccupation with prototyping and user-participation approaches which is found in more recent literature.

In the UK, Hornby et al. (1992) found that most organisations used traditional or structured methodologies (the study did not distinguish between the two) for systems development. Hopker (1994), in a survey of 89 Welsh organisations found that structured methodologies were used by only 33%, less than the 43% using prototyping approaches (however, traditional, waterfall-model approaches were used by 75% of her sample and many of these may have been using structured decomposition approaches, if not the complete, formal methodology).

Saarinen (1990) used the classification framework discussed in the previous section in a study of 21 large, Finnish companies. Two-thirds of Saarinen's sample used linear strategies for system development, indicating traditional or structured approaches. 58% of the sample used third-generation languages to produce the software, with 21% using application generators and 21% using software packages. The average rating on a seven-point scale for formality of the method and management control were 3.98 and 4.46 respectively. This gives a picture very like the US and the UK: showing an overriding concern for process standardisation and hence control (the main benefit of structured and traditional methodologies) rather than support for the creative, communication and learning processes of system stakeholders and software developers.

In both the UK and the US, the majority of those sites which do not use traditional or structured development methodologies use a totally unstructured development process, with no overall methodology (Boehm, 1988; Avison & Fitzgerald, 1988). Boehm (1988) refers to these approaches as "code and fix". Compared with this alternative, structured methodologies, even with their lack of human-centredness, look attractive!

A survey of senior IT managers in 49 large UK organisations was performed in 1995 (Gasson & Holland, 1995)[1]. Of those 32 respondents who performed their software development in-house (and were therefore able to report on development approaches), it was found that only a small minority of sites used 'alternative' systems development methodologies, such as evolutionary prototyping, or approaches to support user-participation. The survey findings are represented graphically in Figure 2.

65% of the sample used the same methodological approach for all three stages of the SDLC. 85% of the sample used an approach in the same group (using the classification of figure 2) for all three stages. The majority of the reported approaches were primarily concerned with

[1] The study of development approaches was performed as part of a larger survey, by Transition Partnerships, of senior IT, functional and finance managers. Copies of the full report can be purchased from Transition Partnerships, Hernshaw, Knowle Lane, Cranleigh, Surrey GU6 8JH, *Tel:* 01483-278452

standardising the process (structured methodologies, computer-aided software engineering (CASE) and internal change control). The non-use of any formal methodological approach was fairly high, but the category "none" may be over-reported as it is possible that the senior IT manager did not know what methods were used in detail. In later stages of the SDLC, there appears to be a slight switch to that group of approaches concerned with speed or automation of system/program generation - fast-build tools, such as the use of 4GLs in Rapid Application Development, CASE or DataBase Management Systems. These approaches might be selected because more organisations are building system prototypes (perhaps for experimental purposes) or it might be that projects increasingly suffer from time-scale pressures as development proceeds.

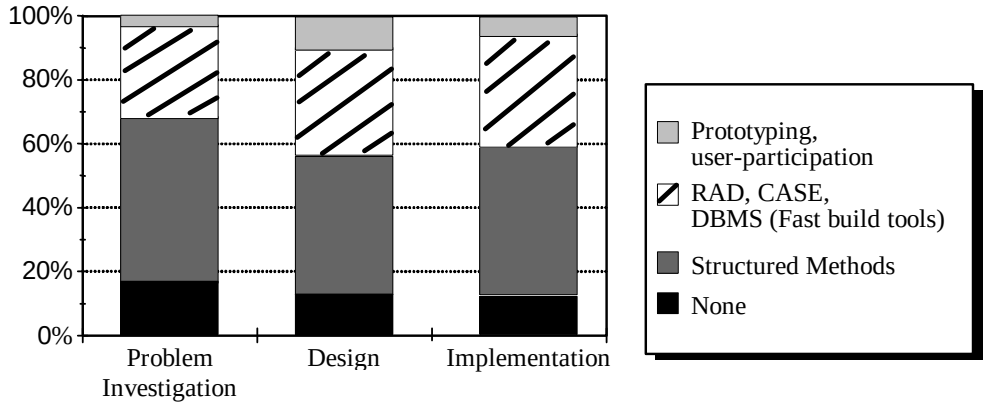

Figure 2: Primary Methodology Used For System Development

These findings show a high use of automation and fast-build approaches. A large proportion of those papers and journal articles aimed at practitioners discuss merits and problems with the CASE approach and, as shown by the survey above, CASE-based methodologies have a significant share of the market. However, consideration of the diffusion of fast-build approaches is largely missing from academic literature. This is possibly because of the preoccupation with prototyping and users discussed earlier, which leads to methodologies being classified into user-participatory or non-user participatory (a classification which sidelines automation of the process).

The overall picture, then, is that the methodologies in use, regardless of national culture, pertain mainly to process standardisation (and hence control) and resource management, rather than the support of team communications, learning, user-involvement or product quality assessment. There appears to be a significant interest in automating software production processes through CASE, but its impact is difficult to gauge as most studies of systems development investigate aspects of the systems development process which are not automated, such as user-requirements elicitation. Although the "software crisis" may be deemed to be over by academics, the increasing use of automated tools may indicate that it looms large in management consciousness.

How Are Methodologies Selected?

Kumar and Bjorn-Andersen (1990) state that the prescription of a particular methodology incorporates into the design process "the ontological assumptions about what constitutes reality and the epistemological assumptions about how to conduct the ISD enquiry." This methodological determinism is challenged by Markus & Bjorn-Andersen (1987) who argue that designers' existing value systems influence the selection of development methodologies: IS development approaches are largely based upon a waterfall model and emphasise

technical/functional optimisation because technical expertise is the basis of IS professionals' power.

If designers' value-systems are formed by the use of a particular methodology, one would expect a very wide variety of IS development cultures in practice, depending upon the methodology in use. This was not found to be so in the studies which examine this aspect. Hornby et al. (1992) found that information systems development is viewed primarily within organisations as a technical problem and therefore approached using technical, function-oriented approaches. This finding is supported by Gasson & Holland (1995). It must be said that there is an element of 'chicken-and-egg' about this debate: do these findings reflect the impact of technically-oriented methodologies upon designers' value-systems, or do they reflect the impact of designers' value systems upon the selection of methodologies?

There is no clear consensus on why certain methodologies may be selected for use. Saarinen's (1990) study started from the position that a particular methodology might be selected according to contingency factors: the clarity of the system 'problem', the level of complexity, the level of uncertainty, the size of the project, the familiarity of the technology etc. However, no discernible link was found between any of the contingency factors and the type of methodology selected: Saarinen concludes that extraneous factors or company standards may have had more influence on the choice of methodology than a rational consideration of alternatives based upon the requirements of the project. Hopker (1994) also argued that it would be rational to use a contingency approach to methodology selection, based upon a strategic classification of the target application. She found little evidence that this approach was used by organisations in practice; selection of methodologies was dominated by historical influences, with a "pick and mix" approach prevailing.

Rosenbrock (1981) suggests that engineers and designers learn a normative approach to problem-solving through the group processes of education and negotiation provided by the project design team, which engenders a positivist and anti-humanist approach to the design of technology. Research on design processes suggests that designers construct a 'design schema': a mental construct which encompasses an understanding of the processes of design and of acceptable solutions for certain types of problems. This schema is derived from the designer's background, education and previous experience of the application domain (c.f. Jeffries et al., 1981; Guindon et al., 1987). If schemas are formed through the normative influences of working in project teams, then it is unsurprising that the selection of development methodologies is based primarily on historical influences: developer experience and methodology reinforce each other in a vicious (or virtuous) circle.

Many developers under-report their use of methodologies: developers are often unaware of the provenance of many of the techniques which they use and are resistant to change (Neccho, 1989). There appears to be a widespread lack of awareness of alternative development approaches and developers acquire their knowledge about available methodologies through informal means, such as periodicals, seminars and vendor training; formal training plans and budgets are non-existent (Neccho, 1989). If developer training is largely achieved through normative learning, this means that many of the approaches used by a developer will necessarily be an amalgam of the various approaches used by teams with which they have worked previously, rather than a rationally-selected methodology, used in its intended philosophical context. Empirical studies by Curtis et al. (Curtis et al., 1988; Curtis & Walz, 1990; Curtis, 1992) indicate the importance of the 'expert designer': an experienced team member who spends most of his or her time educating more junior team members in issues relevant to the application domain and the processes of development. The expert designer can often have a great deal of

influence over the analytical techniques in use by a particular team, sometimes more influence than senior IS managers.

Contingency approaches may not be relevant in the selection of CASE tools. Orlikowski (1993) found that IS developers believed that CASE tools helped them to appear more productive and hence more valuable to their employer. IS managers valued CASE because it allowed them to perform system integration across the organisation with fewer resources and faster than they could otherwise have done, but there was resistance from business managers, who saw CASE as a mechanism for expanding the power of the IT department. Because CASE provided a rapid (and largely implicit) justification to change both corporate data structures and ownership and systems development processes, it provided an effective mechanism to implement IS managers' intentions, whether these were to take control by introducing radical change or to share control by supporting incremental organisational or product change.

The type of methodology selected is not the only factor in determining how development proceeds in an organisation. Many completely different methodological approaches may yield similar cultures, benefits or problems in practice. For example, users may be permitted to participate to a high degree in system development projects which use structured methodologies (Hardy et al., 1994) and may be excluded from system development which uses evolutionary prototyping approaches (Gasson, 1995). Whilst participatory approaches concentrate upon user appreciation and determination of the operation and limitations of new systems, prototyping approaches which concentrate upon *developer* appreciation and determination of the operation and limitations of new systems may lead to unintended user emancipation, through user-evaluation of prototypes. Different methodological approaches may share many of the same objectives and may use common tools and methods. A classification of the methodology is therefore insufficient in determining its effect upon the development process: it is necessary to look at the approach in a much wider sense.

To What Extent Do Methodologies Support Systems Development?

The development methodology has been demonstrated to be important in two respects:

1. The methodology facilitates standardisation and hence management control of the development process, decreasing individuals' autonomy and discretion in design decisions.
2. The methodology embodies the values of technical development staff reinforcing and propagating those values through the normative processes of design (whether or not it *creates* those values).

To these two elements, a third element must now be added:

3. The methodology has a direct impact on the nature, structure and content of system users' jobs.

Traditional methodologies follow the principles of scientific management and consequently tend to produce highly structured and fragmented organisational procedures (Markus and Bjorn-Andersen, 1987; Corbett et al, 1991), while evolutionary methodologies permit users to modify this impact (Eason, 1982). Structured-decomposition methodologies were applied to system development because they involved a "direct control approach", the purpose of which was to make the labour process more stable, predictable and thus more controllable, not because they supported the creative processes of systems development (Jenkins et al., 1984; Friedman and Cornford, 1989). The prevalence of the scientific management approach of structured methodologies would be reason for concern for those who believe that system development

should reflect organisational, business and emancipatory concerns, except that there is evidence (c.f. Sumner & Sitek, 1986; Hardy et al., 1994) to show that structured methodologies are not used in their entirety. Curtis et al. (1988) found that use of a development methodology was often abandoned because of pressures due to management deadlines or commercial and marketing requirements.

The use of structured methodologies may actually be counter-productive. Saarinen (1990) indicates that linear development strategies (based on the waterfall process-model) are more appropriate for large projects, especially where the development problem is relatively well-defined (when requirements are "easy to specify" and uncertainty is low). An experimental study by Boehm et al. (1984) showed that products produced using prototyping methodologies were easier to learn and use than those produced using structured approaches. Structured methodologies were only of use on larger projects, where communications and intersubjective understanding among the design team were more critical to successful completion than the usability of the product. The linear life-cycle model is inaccurate - real-life system development is much more iterative (and sometimes recursive) than the linear model permits (Boehm, 1988).

There is also evidence that structured methodologies also do not provide support for the real-life problem-solving strategies used by system developers. Whilst information system design is represented by structured methodologies as a top-down, decomposition process, empirical studies (Malhotra et al., 1980; Guindon, 1990; Visser & Hoc, 1990) show design to be a process of convergence between a mental model of the proposed solution held by the designer and the set of system requirements, which are re-framed when they cannot be reconciled with solutions available to the designer. Real-life system design is iterative and often recursive.

While Jenkins et al. (1984) reported that project leaders saw systems development methodologies as being of high value for the project, Sumner & Sitek (1986) reported that structured methodologies were not being widely *used* in actual systems development projects because of lack of acceptance by IS professionals who perceived them as time-consuming to use. Hopker's (1994) analysis would also appear to find that IS professionals perceive structured methodologies as having too many techniques and that typical projects are too short to use the full methodology - in other words, that they were too time-consuming to use fully, although she found widespread agreement that they improved process outcomes, such as productivity and specification adherence.

Despite these problems, methodologies are useful in that they maintain intersubjectivity during development processes: Flor & Hutchins (1991) discuss the importance of external structured representations in maintaining a shared, coherent view of the target system among development team members. Too high a level of intersubjectivity and not enough system alternatives are explored for the design to be effective; too low a level and conflict ensues in defining system objectives.

Given the partial use discussed above, a description of the primary methodology does not provide a sufficiently detailed picture of the context of IS development. There is a need for a bridging framework between theory and practice (Keen, 1987) which assesses the approach to information systems development and enables an analysis of an organisation's objectives and philosophical position with respect to their development approach, or enables a comparison of the actual development approach to that intended by the methodology, so that appropriate methodologies may be determined for the approach at each stage of the systems development life-cycle (SDLC). An approach framework was used by Gasson & Holland (1995), in their

survey of organisations' development approach; their overall findings are shown mapped onto this framework in Figure 3.

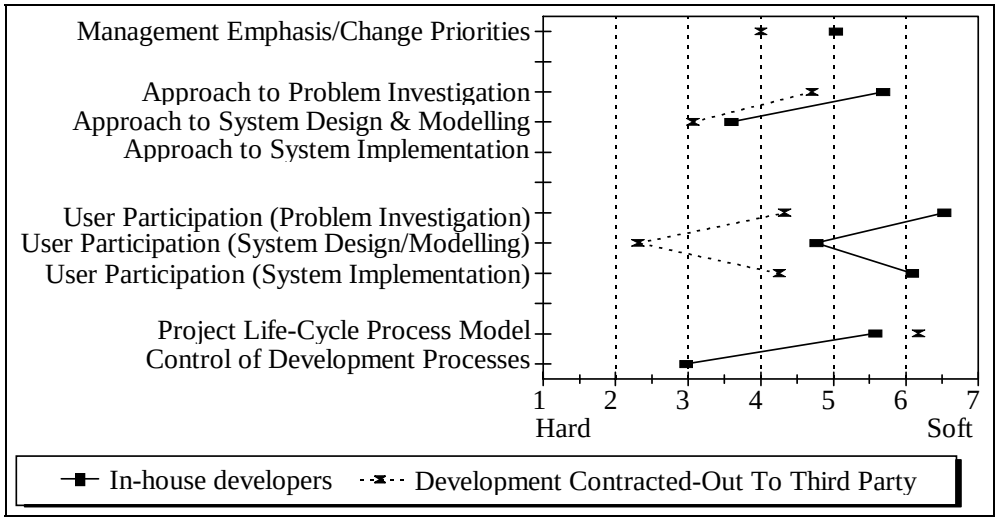

Figure 3: Approaches To IS Development (Gasson & Holland, 1995)

A *coherent* methodological approach is underpinned by a consistent philosophical basis. One would expect organisations using structured methodologies to be mapped firmly along the left-hand axis: given that over 30% of organisations who performed development in-house reported the use of structured methodologies and that the majority of approaches reported were on the 'hard' side of the spectrum, the swings between hard and soft aspects of the development approach would indicate that methodologies are not selected to support a particular approach to development, but for other reasons, such as management control.

Most organisations in the sample saw their overall change priorities and their approach to problem investigation as relatively soft, but for all companies, the approach to system design and modelling was appreciably harder and more technically-oriented. Although the majority of companies reported using the same type of development *methodology* at all stages of the SDLC, their *approach* to system development varied dramatically at different stages. An analysis of development methodologies in use does not tell the whole story: there is a "pick and mix" approach to development which supports a certain approach. This approach will be a negotiated outcome, depending upon the relative power of technical developers, potential system users, functional and technical managers and other organisational stakeholders. It would appear that development is driven by a contingent adoption of various analytical techniques rather than by the philosophical underpinning of a coherent development methodology and selection of methodologies is underpinned more by a desire for control of the process than the support of system design.

Conclusion: The Role of Methodologies In IT-Related Organisational Change

It would appear that the main motivation for selection of a development methodology is control over the development process and that selection is largely steered by IS professionals as an interest group. Within the organisation, although there is a perception that the emphasis of IT-related change is primarily oriented towards organisational change, design of the supporting system is still seen as a technical process, where technical infrastructures, techniques and control mechanisms take precedence over organisational stakeholder participation.

Methodologies based upon a linear systems development life-cycle still appear to be the most popular, possibly for historical reasons. Many organisations use a contingency approach to method selection, using tools from different methodologies according to the task in hand. Automation would appear to be a growing trend, although there is little research evidence to gauge the extent of this trend. However, unlike other methodologies, which may be selected and used on an historical or a local, contingency basis, CASE tools may be selected explicitly to improve the IS manager's control over a wider range of IT-related factors across the organisation.

Methodologies appear not to be used in full, but piecemeal, with little awareness on the part of IS professionals of the provenance of the analytical techniques in use. Formal, structured methodologies are perceived by IS developers as time-consuming and inappropriate to the process of design which lies at the centre of systems development. They are attractive to IS managers and project leaders because of the control which they appear to give over the process. It is possible that methodologies are used as a displacement activity by IS developers and that the real processes of design take place in a much less visible way.

Klein & Hirschheim (1987) discuss a change taking place in the prevailing paradigm of Information Systems research, from a positivist perception of IT-related change as technical/functional ('hard') analysis and design, to a wider perception of social and organisational change. Our requirements of system development methodologies are likely to reflect this change: the objects of development are likely to be wider in scope and more complex than those thought appropriate for well-defined, technical development approaches. However, current research is prioritising some issues, such as user-participation and development project communications and learning, at the expense of others, such as process automation and the creation of organisational data-infrastructures.

Whilst much literature appears to criticise a development methodology for limitations in scope, little of the literature explicitly considers what should be covered by the scope of a methodology. Individual issues are raised, but there is little placing of these issues in a wider context. What is required is a general academic and practitioner debate about what a methodology should be and what role it should fill, against which criteria methodologies may be properly evaluated.

References

Argyris, C. & Schon, D. (1978), Organizational Learning: A Theory Of Action Perspective, Addison-Wesley, Reading, Mass.

Avison, D.E. (1990) 'An Overview of Information Systems Development Methodologies', in R.L. Flood, M.C. Jackson, & P. Keys (Eds.) Systems Prospects: The Next Ten Years of Systems Research, Plenum Press, New York

Avison, D.E. and Fitzgerald, G. (1988) Information Systems Development: Methodologies, Techniques And Tools, Blackwell Scientific Publications, Oxford

Avison, D.E. & Wood-Harper, A.T., (1991), 'Information Systems Development Research: An Exploration of Ideas In Practice', The Computer Journal, 34, 2

Boehm, B.W. (1988), 'A Spiral Model Of Software Development And Enhancement', IEEE Computer Journal, May 1988, 61-72

Boehm, B.W., Gray, T.E. & Seewalt, T. (1984) 'Prototyping Versus Specifying: A Multiproject Experiment', IEEE Transactions on Software Engineering, SE-10, 3

Checkland, P. (1981) Systems Thinking, Systems Practice, Wiley.

Corbett, J.M., Rasmussen, L.B. & Rauner, F. (1991), Crossing the Border: The social and engineering design of computer integrated manufacturing systems, Springer-Verlag, London

Curtis, B. (1992) 'Insights From Empirical Studies Of The Software Design Process', Future Generation Computer Systems 7, 139-149, North Holland, Elsevier

Curtis, B., Krasner, H. and Iscoe, N. (1988), 'A Field Study Of The Software Design Process For Large Systems', Communications of the ACM, 31, 11, 1268-1287

Curtis, B., Walz, D. (1990) 'The Psychology of Programming in the Large: Team and Organizational Behaviour' in J.M. Hoc, T.R.G. Green, R. Samurçay, D.J. Gilmore (Eds.), Psychology of Programming, Academic Press, London, UK

Davidson, E.J. (1993) 'An Exploratory Study of Joint Application Design (JAD) In Information Systems Delivery', in Proceedings of 14th International Conference on Information Systems, ICIS, USA

Eason, K.D. (1982) 'The Process Of Introducing Information Technology', Behaviour and Information Technology, 1, 2, April-June 1982

Flor, N.V. & Hutchins, E.L. (1991) 'Analyzing distributed cognition in software teams: a case study of team programming during perfective software maintenance', in J. Koenemann-Belliveau, T. Moher & S. Robertson (Eds.) Empirical Studies of Programmers - Fourth Workshop, Norwood, NJ: Ablex

Friedman, A.L. and Cornford, D.S. (1989), Computer Systems Development: History, Organisation and Implementation, John Wiley & Sons

Gasson, S. (1995) 'User Involvement In Decision-Making In Information Systems Development' in Proceedings of IRIS 18, August 1995, Gjern, Denmark

Gasson, S. & Holland, N. (1995) 'The Nature And Processes Of IT-Related Change' in Proceedings of IFIP WG8.2 Conference, December 1995, Cambridge, UK

Goldkuhl, G. & Rostlinger, A. (1993) 'Joint Elicitation of Problems: Important Aspects of Change Analysis', in D. Avison, J. Kendall, J. DeGross, Human, Organizational and Social Dimensions of Information Systems Development, IFIP North Holland

Guindon, R. (1990) 'Designing the design process: Exploiting opportunistic thoughts', Human-Computer Interaction, Issue 5, 305-344

Guindon, R., Krasner, H. & Curtis, B. (1987) 'Breakdowns and processes during early activities of software design by individuals', in G.M. Olsen, S. Sheppard, E. Soloway (Eds.) Empirical Studies of Programmers: Second Workshop, Ablex

Hardy, C., Thompson, B. & Edwards, H. (1994) 'Method Use and Customisation in the UK: A Preliminary Survey', in C. Lissoni et al. Proceedings of the Conference of the BCS Information Systems Methodologies Specialist Group, Heriot-Watt University, UK

Hopker, O. (1994) 'Evaluation and Choice of Information Systems Methodologies - A Welsh Perspective', in C. Lissoni et al. Proceedings of the Conference of the BCS Information Systems Methodologies Specialist Group, Heriot-Watt University, UK

Hornby, P., Clegg, C.W., Robson, J.I., Maclaren, C.I., Richardson, S.C., O'Brien, P. (1992) 'Human and Organizational Issues In Information Systems Development', Behaviour & Information Technology, 11, 3

Jayaratna, N. (1994) Understanding and Evaluating Methodologies, A Systemic Framework, McGraw Hill, 1994

Jeffries, R., Turner, A.A., Polson, P.G. & Atwood, M.E. (1981), 'The Processes Involved In Designing Software', in J.R. Anderson (ed.) Cognitive Skills And Their Acquisition, Lawrence Erlbaum Associates, Hillsdale, New Jersey

Jenkins, A., Naumann, J. & Wetherbe, J. (1984) 'Empirical Investigation of Systems Development Practices and Results', Information and Management, 7, 73-82

Keen, P.G.W. (1987) 'MIS Research: Current Status, Trends and Needs', in R.A. Buckingham, R.A. Hirschheim, F.F. Land, and C.J. Tully, Information Systems Education, BCS, Cambridge

Klein, H.K. & Hirschheim, R. (1987) 'Social Change And The Future of Information Systems Development' in R.J. Boland, and R.A. Hirschheim, Critical Issues In Information Systems Research, Wiley.

Kumar, K. & Bjorn-Andersen, N. (1990), 'A Cross-Cultural Comparison Of IS Designer Values', Communications Of The ACM, May 1990, 33, 5, 528-538

Land, F. and Hirschheim, R. (1983) 'Participative Systems Design: Rationale, Tools and Techniques', Journal Of Applied Systems Analysis, 10, 91-107.

Lyytinen, K. (1987) 'A Taxonomic Perspective Of Information Systems Development Theory', in R.J. Boland, and R.A. Hirschheim, Critical Issues In Information Systems Research, Wiley.

Maddison, R., Baker, G., Bhabuta, L., Fitzgerald, G., Hindle, K., Song, J., Stokes, N. & Wood, J. (1984) 'Feature Analysis of Five Information Systems Methodologies', in T. Bemelmans (Ed.) Beyond Productivity: Information Systems Development For Organizational Effectiveness, Elsevier Science Publishers, North Holland

Malhotra, A., Thomas, J., Carroll, J. & Miller, L. (1980) 'Cognitive Processes In Design', International Journal of Man-Machine Studies, 12, 119-140

Markus, M.L. & Bjorn-Andersen, N. (1987), 'Power over users: its exercise by system professionals', Communications of the ACM, June 1987, 30, 6, 498-504

Mumford, E. (1983), Designing Participatively, Manchester Business School, UK

Necchо, C.R. (1989) 'Evaluating Methods of Systems Development: A Management Survey', Journal of Information Systems Management, 6, 1, 8-16
Olle, T.W., Sol, H.G. & Verrijn-Stuart, A.A. (1982), Information Systems Design Methodologies: A Comparative Review, North-Holland, Amsterdam
Olle, T.W., Sol, H.G. & Tully, C.J. (1983), Information Systems Design Methodologies: A Feature Analysis, North-Holland, Amsterdam
Olle, T.W., Sol, H.G. & Verrijn-Stuart, A.A. (1986), Information Systems Design Methodologies: Improving The Practice, North-Holland, Amsterdam
Orlikowski, W.J. (1993) 'CASE Tools as Organizational Change: Investigating Incremental and Radical Changes in Systems Development', MIS Quarterly, September 1993, 309-340
Rosenbrock, H.H. (1981) 'Engineers And The Work That People Do', IEEE Control Systems Magazine, 1, 3 4-8
Saarinen, T. (1990) 'System Development Methodology and Project Success', Information and Management, 19, 3, 183-193
Sumner, M. & Sitek, J. (1986) 'Are Structured Methods for Systems Analysis and Design Being Used?', Journal of Systems Management, June 1986, 37, 6, 18-23
Visser, W. & Hoc J-M. (1990) 'Expert Software Design Strategies' in J.M. Hoc, T.R.G. Green, R. Samurçay, D.J. Gilmore (Eds.), Psychology of Programming, Academic Press, London, UK
Wynekoop, J.L. & Russo, N.L. (1993) 'System Development Methodologies: Unanswered Questions and the Research-Practice Gap', Proceedings of 14th International Conference on Information Systems, ICIS, USA